Conjunctive Queries over Trees*

Georg Gottlob and Christoph Koch
DBAI, TU Wien
A-1040 Vienna, Austria
{gottlob,koch}@dbai.tuwien.ac.at

Klaus U. Schulz
CIS, LMU München
D-80536 München, Germany
schulz@cis.uni-muenchen.de

ABSTRACT

We study the complexity and expressive power of conjunctive queries over unranked labeled trees, where the tree structures are represented using "axis relations" such as "child", "descendant", and "following" (we consider a superset of the XPath axes) as well as unary relations for node labels. (Cyclic) conjunctive queries over trees occur in a wide range of data management scenarios related to XML, the Web, and computational linguistics. We establish a framework for characterizing structures representing trees for which conjunctive queries can be evaluated efficiently. Then we completely chart the tractability frontier of the problem for our axis relations, i.e., we find all subset-maximal sets of axes for which query evaluation is in polynomial time. All polynomial-time results are obtained immediately using the proof techniques from our framework. Finally, we study the expressiveness of conjunctive queries over trees and compare it to the expressive power of fragments of XPath. We show that for each conjunctive query, there is an equivalent acyclic positive query (i.e., a set of acyclic conjunctive queries), but that in general this query is not of polynomial size.

1. INTRODUCTION

The theory of conjunctive queries over relational structures is, from a certain point of view, the greatest success story of the theory of database queries. These queries correspond to the most common queries in database practice, e.g. SQL select-from-where queries with conditions combined using "and" only. They are surprisingly well-behaved: Many important properties hold for conjunctive queries but fail for more general query languages (cf. [5, 1, 18]).

Unranked labeled trees are a clean abstraction of HTML, XML, and LDAP. This motivates the study of the special case of conjunctive queries *over trees*, where the tree structures are represented using unary node label relations and binary relations (often referred to as *axes*) such as *Child*, *Descendant*, and *Following*.

XML Queries. Conjunctive queries over trees are naturally related to the problem of evaluating queries (e.g., XQuery or XSLT) on XML data (cf. [8]). However, conjunctive queries are a cleaner and simpler model whose complexity and expressiveness can be formally studied (while XQuery and XSLT are Turing-complete).

(Acyclic) conjunctive queries over trees are a generalization of the most frequently used fragment of XPath. For example, the XPath query //A[B]/following::C is equivalent to the (acyclic) conjunctive query

$$Q(z) \leftarrow A(x),\ \mathit{Child}(x,y),\ B(y),\ \mathit{Following}(x,z),\ C(z).$$

While XPath has been studied extensively (see e.g. [14, 15] on its complexity, [3, 23] on its expressive power, and [16] on the satisfiability problem), little work so far has addressed the theoretical properties of *cyclic* conjunctive queries over trees. Sporadic results on their complexity can be found in [21, 12, 13, 20].

Data extraction and integration. (Cyclic) conjunctive queries on trees have been used previously in data integration, where queries in languages such as XQuery were canonically mapped to conjunctive queries over trees to build upon the existing work on data integration with conjunctive queries [8, 9]. Another application is Web information extraction using a datalog-like language over trees [2, 13]. (Of course, each nonrecursive datalog rule is a conjunctive query.)

Queries in computational linguistics. A further area in which such queries are employed is computational linguistics, where one needs to search in, or check properties of, large corpora of parsed natural language. Corpora such as Penn Treebank [17] are unranked trees labeled with the phrase structure of parsed (for Treebank, financial news) text. A query asking for prepositional phrases following noun phrases in the same sentence can be phrased as the conjunctive query

$$Q(z) \leftarrow S(x),\ \mathit{Descendant}(x,y),\ \mathit{NP}(y),\ \mathit{Descendant}(x,z),\ \mathit{PP}(z),\ \mathit{Following}(y,z).$$

Figure 1 shows this query in the intuitive graphical notation that we will use throughout the paper (in which nodes correspond to variables, node labels to unary atoms, and edges to binary atoms).

Dominance constraints. Another important issue in computational linguistics are conjunctions of *dominance con-*

*This work was partially supported by project Z29-N04 of the Austrian Science Fund (FWF), by a project of the German Research Foundation (DFG), and by the REWERSE Network of Excellence of the EU.

PODS 2004, June 14–16, 2004, Paris, France.

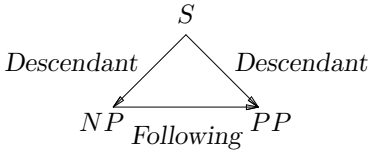


Figure 1: A query graph.

straints [19], which turn out to be equivalent to (Boolean) conjunctive queries over trees. Dominance constraints have been influential as a means of incompletely specifying parse trees of natural language, in cases where (intermediate) results of parsing and disambiguation remain ambiguous. One problem of practical importance is the rewriting of sets of dominance constraints into equivalent but simpler sets (in particular, so-called *solved forms* [4], which correspond to acyclic queries). This implies that studying the expressive power of conjunctive queries over trees, and the problem of deciding whether there is a set of acyclic conjunctive queries equivalent to a given conjunctive query, is relevant to computational linguistics.

Higher-order unification. The query evaluation problem for conjunctive queries over trees is also closely related to the *context matching problem* [25] in higher-order unification[1]. Some tractability frontier for the context matching problem is outlined in [26]. However, little insight is gained from this for the database context, since the classes studied in [26] become unnatural when formulated as conjunctive queries[2].

Contributions

Given the substantial number of applications that we have hinted at above and the nice connection between database theory, computational linguistics, and term rewriting, it is surprising that conjunctive queries over trees have never been the object of a concerted study[3].

In particular, three questions seem worth studying:

1. The complexity of (cyclic) conjunctive queries on trees has only been scratched in the literature. There is little understanding of how the complexity of conjunctive queries over trees depends on the relations used to model the tree.

2. There is a natural connection between conjunctive queries and XPath. Since all XPath queries are acyclic, the question arises whether the acyclic positive queries (i.e., unions of conjunctive queries) are as expressive as the full class of conjunctive queries over trees.[4]

3. If that is the case, how much bigger do the acyclic versions of queries get than their cyclic counterparts? Except from being of theoretical interest, first translating queries into their acyclic versions, if that is possible, and then evaluating them as such may be a practical query evaluation strategy, because there are particularly good algorithms for evaluating such queries [29, 6, 11, 13].

We thus study conjunctive queries on tree structures represented using the XPath *axis* relations *child*, *descendant*, *descendant-or-self*, *following-sibling*, and *following*.[5] For a more principled *symmetric* framework, we study the axes *Child*, $Child^+$ (= *descendant*), $Child^*$ (= *descendant-or-self*), *NextSibling*, $NextSibling^+$ (= *following-sibling*), $NextSibling^*$, and *Following*. (*NextSibling* and $NextSibling^*$ are not supported in XPath but are nevertheless considered here.)

The main contributions of this paper are as follows.

- We establish a framework for proving the tractability of the conjunctive query evaluation problem for a signature τ defining trees based on the new notion of *$<$-hemichordal relations*. The precise definition of $<$-hemichordality is technical. The idea is that of "guarding" such a relation R using a total order $<$ on the elements of the structure (i.e., nodes of the tree) and requiring R to satisfy a weak condition somewhat reminiscent of chordality in graphs. If all relations in τ are $<$-hemichordal (which can be shown easily and independently for each binary relation in τ), the query can be evaluated in polynomial time on all trees of signature τ. The $<$-hemichordality of τ implies that queries can be evaluated by eliminating locally inconsistent solutions. This can be done efficiently. Here, our techniques are reminiscent of work related to *arc-consistency* in *constraint satisfaction* (cf. e.g. [7]).

  Our framework is not restricted to trees, but was not generalized from signatures with unary and binary relations here in order not to weaken its intuitive appeal.

- We determine the complexity of conjunctive queries on trees represented by axis relations and provide a complete characterization of the tractability frontier of the problem (under the assumption that P $\neq$ NP). The subset-maximal sets of axis relations for which the problem is tractable turn out to be disjoint; they are

  $$\{Child, NextSibling, NextSibling^*, NextSibling^+\},$$

  $$\{Child^*, Child^+\}, \quad \text{and} \quad \{Following\}.$$

  Table 1 shows the complexities of conjunctive queries over structures containing unary relations and either one or two axes. Of course, all problems are in NP because conjunctive queries over arbitrary finite structures are [5]. All NP-hardness results hold already for fixed data trees (query complexity).

  Interestingly, the sufficient condition for tractability yielded by our framework serves to immediately detect all the polynomial cases.

  METATHEOREM 1.1. *Unless $P = NP$, for any*

  $$F \subseteq \{Child, Child^*, Child^+, NextSibling, NextSibling^*, NextSibling^+, Following\},$$

[1] To be precise, the analogy is most direct with ranked trees.

[2] These conjunctive queries require node inequality $\neq$ as a binary relation in addition to the tree structure relations. If $\neq$ is removed, the queries become acyclic. However, it is easy to see that already conjunctive queries using only the inequality relation over a fixed tree of three nodes are NP-complete, by a reduction from Graph 3-Colorability.

[3] Of course, as mentioned above, there are a number of papers that implicitly contain relevant results [21, 20, 16, 26].

[4] This is equivalent to asking whether for all conjunctive queries over trees there exist equivalent positive Core XPath queries [14].

[5] Since we are free to use these relations with any pair of variables of our conjunctive queries (differently from XPath), these five axes render all others, i.e. *parent*, *ancestor*, *ancestor-or-self*, *preceding-sibling*, and *preceding*, redundant. Typed child axes such as *attribute* are redundant with the *child* axis and unary relations in our framework.

	Child	*Child*$^+$	*Child**	*NextSibling*	*NextSibling*$^+$	*NextSibling**	*Following*
Child	in P (4.4 [13])	NP-hard (5.1)	NP-hard (5.1)	in P (4.4 [13])	in P (4.4)	in P (4.4)	NP-hard (5.2)
Child$^+$		in P (4.2 [12])	in P (4.2 [12])	NP-hard (5.7)	NP-hard (5.7)	NP-hard (5.7)	NP-hard (5.3)
Child*			in P (4.2 [12])	NP-hard (5.5)	NP-hard (5.4)	NP-hard (5.6)	NP-hard (5.3)
NextSibling				in P (4.4)	in P (4.4)	in P (4.4)	NP-hard (5.8)
NextSibling$^+$					in P (4.4)	in P (4.4)	NP-hard (5.8)
NextSibling*						in P (4.4)	NP-hard (5.8)
Following							in P (4.3)

Table 1: Complexity results for signatures with one or two axes. Numbers are pointers to relevant theorems.

the conjunctive queries over structures with unary relations and binary relations from F are in P iff there is a total order $<$ such that all axes in F are $<$-hemichordal.

- We study the expressive power of conjunctive queries on trees. We show that for each conjunctive query over trees, there is an equivalent acyclic positive query (APQ) over the same tree relations. The blowup in size of the APQs produced is exponential in the worst case.

  It follows that there is an equivalent XPath query for each conjunctive query over trees, since each APQ can be translated into XPath in linear time.

- Finally, we provide a result that sheds some light at the succinctness of (cyclic) conjunctive queries and which demonstrates that the blow-up observed in our translation is actually necessary. We prove that there are conjunctive queries over trees for which no equivalent polynomially-sized APQ exists.

The structure of the paper is as follows. We start with basic notions in Section 2. Section 3 presents our framework for finding classes of conjunctive queries that can be evaluated in polynomial time. Section 4 contains our polynomial-time complexity results. Section 5 completes our tractability frontier with the NP-hardness results. Finally, in Section 6, we provide our expressiveness results.

2. PRELIMINARIES

Let Σ be a labeling alphabet. Throughout the paper, if not explicitly stated otherwise, we will not assume it to be fixed. An unranked tree is a tree in which each node may have an unbounded number of children. We allow for tree nodes to be labeled with multiple labels. However, throughout the paper, our tractability results will support multiple labels while our NP-hardness and expressiveness results will not make use of them.

We represent trees as relational structures using unary label relations $(Label_a)_{a\in\Sigma}$ and binary relations called axes. For a relational structure $\mathcal{A}$, let $A = |\mathcal{A}|$ denote the finite domain (in the case of a tree, the nodes) and let $||\mathcal{A}||$ denote the size of the structure (see e.g. [10]). We use the binary *axis* relations *Child* (defined in the normal way) and *NextSibling* (where *NextSibling*(v, w) iff w is the right neighboring sibling of v in the tree), their transitive and reflexive and transitive closures (denoted *Child*$^+$, *NextSibling*$^+$, *Child**, *NextSibling**), and the axis *Following* (where *Following*(v, w) iff, when the tree is represented as XML, the end tag of v appears before the start tag of w in the XML text). By *Child*$^+$ = *Descendant*, *Child** = *Descendant-or-self*, and *NextSibling*$^+$ = *Following-sibling*, this set of axes covers the standard XPath axes (cf. [28]).

We consider three well-known total orderings on finite ordered trees. The *pre-order* $\leq_{pre}$ corresponds to a depth first left-to-right traversal of a tree. If XML-documents are represented as trees in the usual way, the pre-order coincides with the *document order*. It is given by the sequence of *opening* tags of the XML *elements* (corresponding to nodes). The *post-order* $\leq_{post}$ corresponds to a bottom-up left-to-right traversal of the tree and is given by the sequence of *closing* tags of elements. Furthermore, we also consider the ordering $\leq_{bflr}$ which is given by the sequence of opening tags if we traverse the tree breadth-first left-to-right.

Boolean (0-ary), monadic (unary), and k-ary conjunctive queries are defined in the normal way (cf. [1]). Let Q be a conjunctive query and let *Var*(Q) denote the variables appearing in Q. The query graph of Q over unary and binary relations is the directed multigraph $G = (V, E)$ with edge labels and multiple node labels such that $V =$ *Var*(Q), node x is labeled P iff Q contains unary atom $P(x)$, and E contains labeled directed edge $x \xrightarrow{R} y$ iff Q contains binary atom $R(x, y)$. Figure 1 shows an example of such a query graph.

Throughout the paper, we use lower case node and variable names and upper case label and relation names.

3. GLOBAL VS. ARC-CONSISTENCY

Let Q be a conjunctive query and let A denote the finite domain, i.e. in case of a tree the set of nodes. A pre-valuation for Q is a total function Θ : *Var*$(Q) \to 2^A$ that assigns to each variable of Q a *nonempty* subset of A. A valuation for Q is a total function θ : *Var*$(Q) \to A$.

Let $\mathcal{A}$ be a relational structure of unary and binary relations. A pre-valuation Θ is called *arc-consistent*[6] iff for each unary atom $P(x)$ in Q and each $v \in \Theta(x)$, $P(v)$ is true (in $\mathcal{A}$) and for each binary atom $R(x, y)$ in Q, for each $v \in \Theta(x)$ there exists $w \in \Theta(y)$ s.t. $R(v, w)$ is true and for each $w \in \Theta(y)$ there exists $v \in \Theta(x)$ s.t. $R(v, w)$ is true.

PROPOSITION 3.1 (FOLKLORE). *There is an algorithm which checks in time $O(||\mathcal{A}|| \cdot |Q|)$ whether an arc-consistent pre-valuation of Q on $\mathcal{A}$ exists, and if it does, returns one.*

[6]This notion is well-known in constraint satisfaction, c.f. [7].

Proof Sketch. We phrase the problem of computing Θ by deciding, for each x, v, whether $v \notin \Theta(x)$ as an instance $\mathcal{P}$ of propositional Horn-SAT. The propositional predicates are the pairs (x, v), and the Horn clauses are

$$\{(x,v) \leftarrow . \mid P(x) \in Q,\ v \in A,\ \neg P^{\mathcal{A}}(v)\} \cup$$
$$\{(x,v) \leftarrow \bigwedge\{(y,w) \mid R^{\mathcal{A}}(v,w)\}. \mid R(x,y) \in Q,\ v \in A\} \cup$$
$$\{(y,w) \leftarrow \bigwedge\{(x,v) \mid R^{\mathcal{A}}(v,w)\}. \mid R(x,y) \in Q,\ w \in A\}$$

The program $\mathcal{P}$ can be computed and solved (e.g. using Minoux' algorithm [22]), and the solution complemented, in time linear in the size of the program, which is $O(||\mathcal{A}|| \cdot |Q|)$. If there is a variable x such that for no node v, (x, v) is in the solution of $\mathcal{P}$, no arc-consistent pre-valuation of Q on $\mathcal{A}$ exists and Q is not satisfied. Otherwise, the pre-valuation defined by $\Theta(x) \mapsto \{v \mid (x, v) \text{ is in the solution of } \mathcal{P}\}$, for each x, is obviously arc-consistent and contains all arc-consistent pre-valuations of Q and $\mathcal{A}$. □

Actually, this algorithm computes the unique subset-maximal arc-consistent pre-valuation of Q on $\mathcal{A}$.

A valuation θ is called *consistent* if it satisfies the query.[7] Obviously, this is true iff the pre-valuation Θ defined by $\Theta(x) \mapsto \{\theta(x)\}$ is arc-consistent. Let $<$ be a total order on $A = |\mathcal{A}|$ and Θ be a pre-valuation. Then the valuation θ with $\theta(x) \mapsto v$ iff v is the smallest node in $\Theta(x)$ w.r.t. $<$ is called the *minimum valuation* w.r.t. $<$ in Θ.

DEFINITION 3.2. Let $\mathcal{A}$ be a relational structure, R a binary relation in $\mathcal{A}$, and $<$ a total order on $A = |\mathcal{A}|$. Then, R is called *$<$-hemichordal* iff for all n_0, n_1, n_2, n_3 s.t. $n_0 < n_1$ and $n_0 \leq n_2 \leq n_3$,

1. $R(n_1, n_2) \wedge R(n_0, n_3) \rightarrow R(n_0, n_2)$ and
2. $R(n_2, n_1) \wedge R(n_3, n_0) \rightarrow R(n_2, n_0)$. □

Let $\mathcal{A}$ be a structure of unary and binary relations and let $<$ be a total order on $A = |\mathcal{A}|$. $\mathcal{A}$ is called $<$-hemichordal if all binary relations R in $\mathcal{A}$ are $<$-hemichordal.

LEMMA 3.3. *Let $\mathcal{A}$ be a $<$-hemichordal structure and let Θ be an arc-consistent pre-valuation on $\mathcal{A}$ for a given conjunctive query over the relations of $\mathcal{A}$. Then, the minimum valuation in Θ w.r.t. $<$ is consistent.*

Proof. Let θ denote the minimum valuation in Θ w.r.t. $<$. We show the following: If α is any binary atom of Q with variables x, y then α holds under assignment θ. Let $\theta(x) = n_0$ and $\theta(y) = n_2$. W.l.o.g., we assume that $n_0 \leq n_2$, and distinguish two cases:

Case 1 (α has the form $R(x, y)$): Since Θ is arc-consistent there exists a node $n_1 \in \Theta(x)$ s.t. $R(n_1, n_2)$ and a node $n_3 \in \Theta(y)$ s.t. $R(n_0, n_3)$. As θ is a minimum valuation and $R(n_1, n_2)$, we have $\theta(x) = n_0 \leq n_1$. Since $R(n_0, n_3)$ is true, we have $\theta(y) = n_2 \leq n_3$. If $n_0 = n_1$ then $R(\theta(x), \theta(y))$ is true and we are done. Otherwise, $n_0 < n_1$, thus it follows from condition 1 of Definition 3.2 that $R(n_0, n_2)$.

Case 2 (α has the form $R(y, x)$): Since Θ is arc-consistent there exists a node $n_1 \in \Theta(x)$ s.t. $R(n_2, n_1)$ and a node $n_3 \in \Theta(y)$ s.t. $R(n_3, n_0)$. As θ is a minimum valuation and $R(n_2, n_1)$, we have $\theta(x) = n_0 \leq n_1$. Since $R(n_3, n_0)$ is true, we have $\theta(y) = n_2 \leq n_3$. If $n_0 = n_1$ then $R(\theta(y), \theta(x))$ is true and we are done. Otherwise, $n_0 < n_1$, thus it follows from condition 2 of Definition 3.2 that $R(n_2, n_0)$. □

Clearly, if no arc-consistent pre-valuation of Q on $\mathcal{A}$ exists, there is no consistent valuation for Q on $\mathcal{A}$.

THEOREM 3.4. *Given a $<$-hemichordal structure $\mathcal{A}$ and a Boolean conjunctive query Q over $\mathcal{A}$, Q can be evaluated on $\mathcal{A}$ in time $O(||\mathcal{A}|| \cdot |Q|)$.*

Proof. By Lemma 3.3, to check whether a Boolean query Q is satisfied, all we need to do is to try to compute the subset-maximal arc-consistent pre-valuation Θ w.r.t. Q. By Proposition 3.1, this can be done in time $O(||\mathcal{A}|| \cdot |Q|)$. If it exists, Q returns true, otherwise, it returns false. □

If follows that checking whether a given tuple $\langle a_1, \ldots, a_k \rangle$ is in the result of a k-ary conjunctive query on $<$-hemichordal structures can be decided in time $O(||\mathcal{A}|| \cdot |Q|)$ as well. All we need to do is to add (new) singleton unary relations $X_1 = \{a_1\}, \ldots, X_k = \{a_k\}$ to $\mathcal{A}$ and to rewrite the query $Q(x_1, \ldots, x_k) \leftarrow \Phi(x_1, \ldots, x_k)$ into the Boolean query $Q \leftarrow \Phi(x_1, \ldots, x_k) \wedge X_1(x_1) \wedge \cdots \wedge X_k(x_k)$. A k-ary conjunctive query Q over $\mathcal{A}$ with $A = |\mathcal{A}|$ can thus be evaluated on $\mathcal{A}$ in time $O(|A|^k \cdot ||\mathcal{A}|| \cdot |Q|)$.

For relations that are subsets of the given total order $\leq$, a simpler condition for $<$-hemichordality can be given.

LEMMA 3.5. *Let $\mathcal{A}$ be a structure, $<$ a total order on $A = |\mathcal{A}|$, and R a binary relation of $\mathcal{A}$ such that $R \subseteq \leq$. Then, R is $<$-hemichordal iff for all $n_0, n_1, n_2, n_3 \in A$,*

$$n_0 < n_1 \leq n_2 < n_3 \wedge R(n_1, n_2) \wedge R(n_0, n_3) \rightarrow R(n_0, n_2).$$

Proof Sketch. Obviously, if the condition for $<$-hemichordality of Definition 3.2 holds for a given R and $<$, the (strictly weaker) condition of our lemma holds as well.

For the other direction, assume that $R \subseteq \leq$ and the condition of our lemma holds. Then the two conditions of Definition 3.2 hold as well: (1) If $R(n_1, n_2)$ is true then $n_1 \leq n_2$ and if $n_2 = n_3$ then $R(n_0, n_3)$ entails $R(n_0, n_2)$, so $R(n_1, n_2) \wedge R(n_0, n_3) \rightarrow R(n_0, n_2)$ is true whenever $n_0 < n_1$ and $n_0 \leq n_2 \leq n_3$. (2) If $R(n_3, n_0)$ is true then $n_3 \leq n_0$, and thus, since $n_0 \leq n_2 \leq n_3$, $n_0 = n_2 = n_3$. But then, $R(n_3, n_0)$ trivially implies $R(n_2, n_0)$. □

REMARK 3.6. Note that Lemma 3.5 extends to the case where $R \subseteq \geq$. If $R \subseteq \geq$ and for all $n_0, n_1, n_2, n_3 \in |\mathcal{A}|$,

$$n_0 < n_1 \leq n_2 < n_3 \wedge R(n_3, n_0) \wedge R(n_2, n_1) \rightarrow R(n_2, n_0),$$

then R^{-1} is $<$-hemichordal by Lemma 3.5. We may replace all atoms of the form $R(x, y)$ in Q by $R^{-1}(y, x)$ without affecting the meaning. □

For total order $<$, let

$$Succ_< := \{\langle x, y \rangle \mid x < y \ \wedge \nexists z\ x < z < y\}.$$

It is trivial to verify that $Succ_<$, $<$, and $\leq$ are $<$-hemichordal.

4. POLYNOMIAL-TIME RESULTS

Lemma 3.5 and the results of the previous section provide us with a simple technique for proving polynomial-time complexity results for conjunctive queries over trees. Indeed, there is a wealth of inclusions of axis relations in the total orders introduced in Section 2: the axes (1) *Child*,

[7] In this case, and for a Boolean query, we also say that the structure is a *model* of the query.

Child$^+$, *Child**, *NextSibling*, *NextSibling*$^+$, *NextSibling**, and *Following* are subsets of the pre-order $\leq_{pre}$, (2) *Child*$^{-1}$, (*Child*$^+$)$^{-1}$, (*Child**)$^{-1}$, *Following*, *NextSibling*, *NextSibling*$^+$, and *NextSibling** are subsets of the post-order $\leq_{post}$, and (3) *Child*, (*Child*$^+$)$^{-1}$, (*Child**)$^{-1}$, *NextSibling*, *NextSibling*$^+$, and *NextSibling** are subsets of the order $\leq_{bflr}$.

Using Lemma 3.5, it is straightforward to show that

THEOREM 4.1. *The axes*

1. *Child*$^+$ *and Child** *are* $<_{pre}$*-hemichordal,*
2. *Following is* $<_{post}$*-hemichordal, and*
3. *Child, NextSibling, NextSibling***, and NextSibling*$^+$ *are* $<_{bflr}$*-hemichordal.*

Proof. All proof arguments use Lemma 3.5.

We first show that *Child** is $<_{pre}$-hemichordal. (The proof for *Child*$^+$ is similar.) Consider nodes $n_0, \ldots, n_3$ s.t. $n_0 <_{pre} n_1 \leq_{pre} n_2 <_{pre} n_3$, *Child*$^*(n_0, n_3)$, and *Child*$^*(n_1, n_2)$. It is simple to see that $\leq_{pre}$ is the disjoint union of *Child** and *Following*. Therefore, either *Child*$^*(n_0, n_1)$, which implies *Child*$^*(n_0, n_2)$, or *Following*(n_0, n_1). The latter case would yield $n_3 <_{pre} n_1$, a contradiction.

Next, we show is that *Following* is $<_{post}$-hemichordal. Assume that $n_0 <_{post} n_1 \leq_{post} n_2 <_{post} n_3$ and *Following*(n_1, n_2), *Following*(n_0, n_3). Clearly, the relation $\leq_{post}$ is the disjoint union of *Following* and the inverse of *Descendant-or-self*. Since $n_0 <_{post} n_1$ is true, either *Descendant-or-self*(n_1, n_0) or *Following*(n_0, n_1) must hold. In both cases it follows that *Following*(n_0, n_2). Thus, *Following* is $<_{post}$-hemichordal.

The fact that *Child* is $<_{bflr}$-hemichordal follows trivially from Lemma 3.5: Assume that $n_0 <_{bflr} n_1 \leq_{bflr} n_2 <_{bflr} n_3$, *Child*$(n_0, n_3)$, and *Child*$(n_1, n_2)$. Since $n_0 <_{bflr} n_1 \leq_{bflr} n_3$ and *Child*(n_0, n_3), there are just the following three cases: (1) n_1 is a left sibling of n_3, (2) $n_1 = n_3$, or (3) n_1 is a right sibling of n_0. In all cases, *Child*$(n_1, n_2) \wedge n_2 \leq_{bflr} n_3$ leads to a contradiction.

It is easy to verify that *NextSibling*, *NextSibling**, and *NextSibling*$^+$ are $<_{bflr}$-hemichordal using Lemma 3.5. □

Now, it follows immediately from Lemma 3.3 that

COROLLARY 4.2 ([12]). *Conjunctive queries over the signature*

$$\tau_1 := \langle (Label_a)_{a\in\Sigma}, Child^+, Child^* \rangle$$

are in polynomial time w.r.t. combined complexity.

COROLLARY 4.3. *Conjunctive queries over the signature*

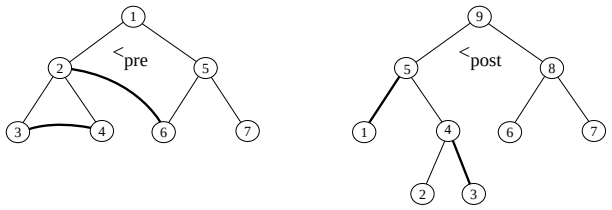
$$\tau_2 := \langle (Label_a)_{a\in\Sigma}, Following \rangle$$

are in polynomial time w.r.t. combined complexity.

COROLLARY 4.4. *Conjunctive queries over the signature*

$$\tau_3 := \langle (Label_a)_{a\in\Sigma}, Child, NextSibling, NextSibling^*, NextSibling^+ \rangle$$

are in polynomial time w.r.t. combined complexity.



(a) (b)

Figure 2: (a) *Following* **is not** $<_{pre}$**-hemichordal; (b)** *Descendant*$^{-1}$ **and** *Descendant-or-self*$^{-1}$ **are not** $<_{post}$**-hemichordal.**

REMARK 4.5. The remaining inclusions between axis relations and total orders introduced at the beginning of this section do not extend to $<$-hemichordality.

The examples in Figure 2 show that (a) *Following* does not satisfy Lemma 3.5 w.r.t. pre-order $<_{pre}$ and that (b) *Descendant*$^{-1}$ and *Descendant-or-self*$^{-1}$ do not satisfy the condition of Remark 3.6 w.r.t. post-order $<_{post}$. □

5. NP-HARDNESS RESULTS

In this section, we study the complexity of the conjunctive query evaluation problem for the remaining sets of axis relations. We are able to show that for all cases for which our techniques based on $<$-hemichordality do not yield a polynomial-time complexity result, we are able to prove NP-hardness. All NP-hardness results hold already for query complexity, i.e., in a setting where the data tree is fixed and only the query is assumed variable.

All reductions are from *one-in-three 3SAT*, which is the following NP-complete [24] problem: Given a set U of variables, a collection $\mathbf{C}$ of clauses over U such that each clause $C \in \mathbf{C}$ has $|C| = 3$, is there a truth assignment for U such that each clause in $\mathbf{C}$ has exactly one true literal? 1-in-3 3SAT remains NP-complete if all clauses contain only positive literals.

Below, we will use shortcuts of the form $\chi^k(x, y)$, where χ is an axis, in queries to denote chains of k χ-atoms leading from variable x to y. For example, *Child*$^2(x, y)$ is a shortcut for *Child*(x, z), *Child*(z, y), where z is a new variable.

The first theorem strengthens a known result for combined complexity [21] to query complexity.

THEOREM 5.1. *Conjunctive queries over the signatures*

$$\tau_4 := \langle (Label_a)_{a\in\Sigma}, Child, Child^+ \rangle$$
$$\tau_5 := \langle (Label_a)_{a\in\Sigma}, Child, Child^* \rangle$$

are NP-complete w.r.t. query complexity.

Proof. Let $C_1, \ldots, C_m$ be a 1-in-3 3SAT instance over positive literals. We assume that C_i is an ordered sequence of three positive literals. We may assume w.l.o.g. that no clause contains a particular literal more than once.

We encode this instance as one of the Boolean conjunctive query evaluation problem for τ_4 (τ_5). The fixed data tree is shown in Figure 3.

For the query, we introduce variables x_i, y_i for $1 \leq i \leq m$ and in addition a variable $z_{k,l,i,j}$ whenever the k-th literal of C_i coincides with the l-th literal of C_j ($1 \leq i \leq m$, $1 \leq j \leq m$, $i \neq j$, $1 \leq k, l \leq 3$).

The Boolean query consists of the following atoms:

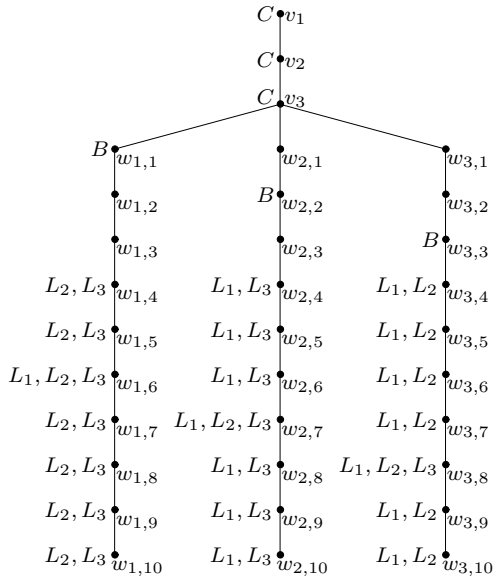


Figure 3: Data tree of the proof of Theorem 5.1.

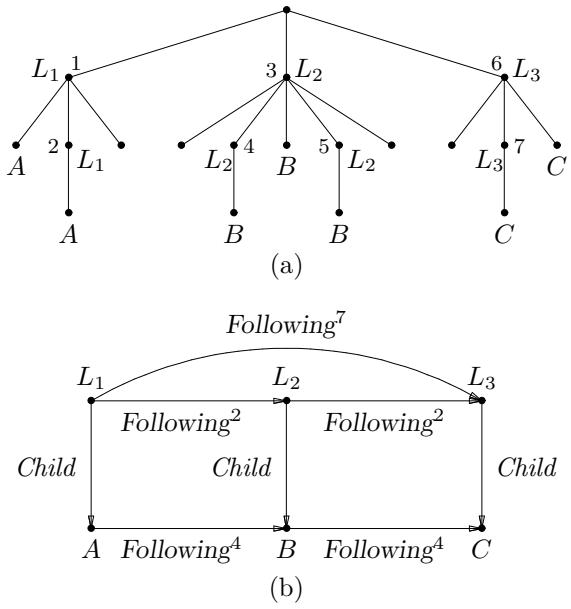


Figure 4: Clause gadget of proof of Theorem 5.2.

- for $1 \leq i \leq m$,

$$C(x_i),\ B(y_i),\ \mathit{Child}^3(x_i, y_i),$$

- for each variable $z_{k,l,i,j}$,

$$L_k(z_{k,l,i,j}), \mathit{Child}^{\circ}(y_i, z_{k,l,i,j}), \mathit{Child}^{8+k-l}(x_j, z_{k,l,i,j})$$

where $\circ$ is "+" on signature τ_4 and "$*$" on τ_5.

"$\Rightarrow$". To prove correctness of the encoding scheme, we first show that given any solution mapping

$$\sigma : \{1, \ldots, m\} \to \{1, 2, 3\}$$

of $C_1, \ldots, C_m$ (i.e., $\sigma(i) = k'$ iff σ selects the k'-th literal from C_i) we can define a satisfaction θ of the query. We first define a valuation θ of our query and then show that all constraints are satisfied. We set

- $\theta(x_i) := v_{\sigma(i)}$ for $1 \leq i \leq m$,
- $\theta(y_i) := w_{\sigma(i),\sigma(i)}$ for $1 \leq i \leq m$, and
- for each variable $z_{k,l,i,j}$, $\theta(z_{k,l,i,j}) := w_{\sigma(i),5+k-l+\sigma(j)}$.

We now prove that θ is a satisfaction of the query. Our choice implies that the variables x_i and y_i are mapped to nodes with labels C and B, respectively. Furthermore, $\theta(y_i) = w_{\sigma(i),\sigma(i)}$ can be reached from $\theta(x_i) = v_{\sigma(i)}$ with three child-steps. For any variable of the form $z_{k,l,i,j}$, $\theta(z_{k,l,i,j}) = w_{\sigma(i),5+k-l+\sigma(j)}$ is always a Child° of $w_{\sigma(i),\sigma(i)}$. If $\sigma(i) \neq k$, then $\theta(z_{k,l,i,j}) = w_{\sigma(i),5+k-l+\sigma(j)}$ has label L_k because $4 \leq 5 + k - l + \sigma(j) \leq 10$ and the nodes $w_{\sigma(i),4}, \ldots, w_{\sigma(i),10}$ all have (at least) the two labels $L_{k'}$ for which $\sigma(i) \neq k'$. If $\sigma(i) = k$, then $\sigma(j) = l$. By going $8 + k - l$ steps downward from $v_{\sigma(j)}$, passing through $w_{k,k}$, we reach node $w_{k,5+k}$, which has label L_k. Since $\theta(z_{k,l,i,j}) = w_{\sigma(i),5+k-l+\sigma(j)} = w_{k,5+k}$, the constraints $\mathit{Child}^{8+k-l}(x_j, z_{k,l,i,j})$ are satisfied. Therefore, θ is indeed a satisfaction of our query.

"$\Leftarrow$". To finish the proof we show that from any satisfaction θ of the query we obtain a corresponding solution for the 1-in-3 3SAT instance $C_1, \ldots, C_m$. If $\theta(x_i) = v_k$, we interpret this as the k-th literal of clause C_i being chosen to be true. Obviously, under any valuation of the query, we select precisely one literal from each clause C_i. We have to verify that if a literal L occurs in two clauses C_i and C_j and we select L in C_i, we also select L in C_j. Let L be the k-th literal of C_i and let $\theta(x_i) = v_k$ (i.e., L is selected in C_i). Then $\theta(z_{k,l,i,j}) = w_{k,5+k}$ because that is the only node below $\theta(y_i) = w_{k,k}$ that has label L_k. The query contains the atom $\mathit{Child}^{8+k-l}(x_j, z_{k,l,i,j})$ for variable $z_{k,l,i,j}$. From node $w_{k,5+k}$, by $8 + k - l$ upward steps we arrive at node v_l. Hence $\theta(x_j) = v_l$, and we select L from clause C_j.

Some nodes in the data tree carry multiple labels. However, since the *Child* axis is available in both τ_4 and τ_5, multiple labels can be eliminated by pushing them down to new children in the data tree and modifying the queries accordingly. □

THEOREM 5.2. *Conjunctive queries over the signature*

$$\tau_6 := \langle (\mathit{Label}_a)_{a \in \Sigma}, \mathit{Child}, \mathit{Following} \rangle$$

are NP-complete w.r.t. query complexity.

Proof Sketch. Consider the construction shown in Figure 4, consisting of the fragment of a data tree (a) and of a query (b) over the labeling alphabet $\Sigma = \{A, B, C, L_1, L_2, L_3\}$.

Observe that the labels L_1, L_2, and L_3 occur only once each in Figure 4 (b). We will refer to the nodes (= query variables) labeled L_1, L_2, and L_3 by v_1, v_2, and v_3, respectively. For the following discussion, we have annotated some of the nodes of the data tree with numbers (1–7). Below, node 1 (resp. 3, 6) is called the *topmost position* of variable v_1 (resp. v_2, v_3). We start with three simple observations.

1. *In any satisfaction θ of the query on the data tree, at most one of the variables v_1, v_2, and v_3 is mapped to its topmost position under θ.* In fact, assume, e.g., that $\theta(v_1) = 1$. From node 1, node 3 (resp. 6) cannot be reached by a sequence of 2 (resp. 7) *Following*-steps. Hence we have $\theta(v_2) \neq 3$ and $\theta(v_3) \neq 6$.

$k\backslash l$	1	2	3
1	10	13	18
2	5	8	13
3	2	5	10

Table 2: The function $NAND(k,l)$.

2. *In any solution θ of the problem, at least one of the variables v_1, v_2, and v_3 is mapped to its topmost position under θ.* In fact, assume that $\theta(v_1) = 2$ and $\theta(v_2) \neq 3$. The constraints in the query (in particular, on the variables corresponding to nodes on the bottom of the query graph) require that $\theta(v_2) \neq 4$. Hence $\theta(v_2) = 5$ is the only remaining possibility. But now the query requires that $\theta(v_3) \neq 7$. Hence $\theta(v_3) = 6$.

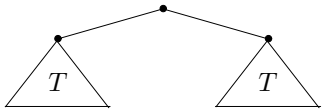
Thus, precisely the three partial assignments

$$(a) \quad \theta(P_i) := 1,\ \theta(Q_i) := 4,\ \theta(R_i) := 7$$
$$(b) \quad \theta(P_i) := 2,\ \theta(Q_i) := 3,\ \theta(R_i) := 7$$
$$(c) \quad \theta(P_i) := 2,\ \theta(Q_i) := 5,\ \theta(R_i) := 6$$

can be extended to a satisfaction of the query. Precisely one of the variables v_1, v_2, and v_3 is mapped to its topmost position under each of the above assignments. Conversely, for each variable there is a satisfying assignment in which it takes its topmost position.

Given a clause C, an *ordered* list of three positive literals, we interpret a satisfaction θ in which variable v_k is mapped to its topmost position as the selection of the k-th literal from C to be true. The encoding described above thus assures that exactly one variable of clause C is selected and becomes true.

Now consider a 1-in-3 3SAT problem instance over positive literals with clauses $C_1, \ldots, C_m$. We encode such an instance as a conjunctive query over τ_6 and a fixed data tree over labeling alphabet $\Sigma = \{A, B, C, L_1, L_2, L_3\}$. This tree consists of two copies of the tree of Figure 4 (a) under a common root, i.e.,



where T denotes the tree of Figure 4 (a).

The query is obtained as follows. Each clause C_i is represented using two copies of the query gadget of Figure 4 (b) (a "left" copy Q_i and a "right" copy Q'_i). We wire the two sets of subqueries $Q_1, \ldots, Q_m, Q'_1, \ldots, Q'_m$ as follows.

Consider first the integer function $NAND(k,l)$ defined by Table 2. We can enforce that two variables, x and y, labeled L_k and L_l in their respective subqueries, cannot both match the topmost node labeled L_k resp. L_l in the left, respective right, part of the data tree by adding a constraint of the form $\mathit{Following}^{NAND(k,l)}(x,y)$ to the query.

For each pair of clauses C_i, C_j, variable x such that Q_i (resp., Q'_i) contains the unary atom $L_k(x)$, and variable y such that Q'_j (resp., Q_j) contains the unary atom $L_l(y)$, if

- the k-th literal of C_i occurs also in C_j and
- the k-th literal of C_i and the l-th literal of C_j are different,

then we add an atom $\mathit{Following}^{NAND(k,l)}(x,y)$ to the query.

These constraints make sure that if a literal is chosen to be true in one clause, it must be selected to be true in all other clauses as well. In the case that $i = j$, the idea is to make sure that both copies of the query gadget of each clause, Q_i and Q'_i, make the same choice of selected literal. The case that $i \neq j$ models the interaction between distinct clauses. The constraints assure that each literal is assigned the same truth value in all clauses.

Using two copies of the query gadget for each clause and two copies of the tree gadget of Figure 4 (a) in the data tree is necessary, as we cannot use $\mathit{Following}^k$-constraints to make sure that two variables are not both assigned their topmost positions in the data tree (corresponding to "true") if the data tree consists just of the tree of Figure 4 (a) and these two topmost positions in the data tree coincide.

This concludes the construction, which can be easily implemented to run in logarithmic space. It is not difficult to verify that the fixed data tree satisfies the query precisely if the 1-in-3 3SAT instance is satisfiable. □

THEOREM 5.3. *Conjunctive queries over the signatures*

$$\tau_7 := \langle (\mathit{Label}_a)_{a\in\Sigma}, \mathit{Child}^+, \mathit{Following}\rangle,$$
$$\tau_8 := \langle (\mathit{Label}_a)_{a\in\Sigma}, \mathit{Child}^*, \mathit{Following}\rangle$$

are NP-complete w.r.t. query complexity.

Proof Sketch. The same encoding as in the previous proof can be used, with the only difference that Child^* resp. Child^+ is used instead of Child in the query. In fact, if the topmost position for v_1 (resp. v_2, v_3) is chosen, there are two possible matches for "A" (resp. three for "B" and two for "C"). This has no impact on the constraints across clauses or the constraints that at most one variable of each clause is assigned to its topmost position. To make sure that *at least* one variable of each clause is assigned its topmost position, the constraints of the query assure that either "A", "B", or "C" are assigned to the correspondingly labeled node at depth two in the subtree of the clause (rather than depth three). □

Since by definition,

$$\mathit{Following}(x,y) = \exists z_1 \exists z_2\ \mathit{Child}^*(z_1,x) \wedge \mathit{NextSibling}^+(z_1,z_2) \wedge \mathit{Child}^*(z_2,y),$$

COROLLARY 5.4. *Conjunctive queries over the signature*

$$\tau_9 := \langle (\mathit{Label}_a)_{a\in\Sigma}, \mathit{Child}^*, \mathit{NextSibling}^+\rangle$$

are NP-complete w.r.t. query complexity.

THEOREM 5.5. *Conjunctive queries over the signature*

$$\tau_{10} := \langle (\mathit{Label}_a)_{a\in\Sigma}, \mathit{Child}^*, \mathit{NextSibling}\rangle$$

are NP-complete w.r.t. query complexity.

Proof Sketch. If we replace *Following* by

$$\mathit{Following}'(x,y) := \exists z_1 \exists z_2\ \mathit{Child}^*(z_1,x) \wedge \mathit{NextSibling}(z_1,z_2) \wedge \mathit{Child}^*(z_2,y),$$

we can reuse the construction of the proof of Theorem 5.2 (in the modified form of the proof of Theorem 5.3). □

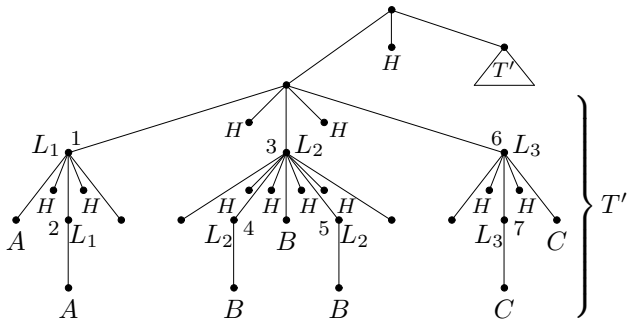


Figure 5: Data tree of proof of Theorem 5.6.

THEOREM 5.6. *Conjunctive queries over the signature*

$$\tau_{11} := \langle \mathit{Child}^*, \mathit{NextSibling}^*, (\mathit{Label}_a)_{a\in\Sigma}\rangle$$

are NP-complete w.r.t. query complexity.

Proof Sketch. The proof basically uses the same argument as Corollary 5.4. However, since we now have to deal with $\mathit{NextSibling}^*$ rather than $\mathit{NextSibling}^+$, we need a way to ensure that $\mathit{NextSibling}^*$ moves at least one step to the right. We thus replace each occurrence of *Following* in the construction of the proof of Theorem 5.2 by

$$\begin{aligned}\mathit{Following}'(x,y) := \exists z_1 \exists z_2 \exists z_3\ \mathit{Child}^*(z_1,x) \wedge \\ \mathit{NextSibling}^*(z_1,z_2) \wedge H(z_2) \wedge \\ \mathit{NextSibling}^*(z_2,z_3) \wedge \mathit{Child}^*(z_3,y).\end{aligned}$$

The modified data tree is as shown in Figure 5. It uses specially labeled auxiliary nodes inserted between each pair of adjacent siblings in the data tree of the proof of Theorem 5.2. □

THEOREM 5.7. *Conjunctive queries over the signatures*

$$\begin{aligned}\tau_{12} &:= \langle \mathit{Child}^+, \mathit{NextSibling}, (\mathit{Label}_a)_{a\in\Sigma}\rangle,\\ \tau_{13} &:= \langle \mathit{Child}^+, \mathit{NextSibling}^+, (\mathit{Label}_a)_{a\in\Sigma}\rangle,\\ \tau_{14} &:= \langle \mathit{Child}^+, \mathit{NextSibling}^*, (\mathit{Label}_a)_{a\in\Sigma}\rangle\end{aligned}$$

are NP-complete w.r.t. query complexity.

Proof Sketch. The proofs are analogous to the proofs for the respective signatures with Child^* rather than Child^+, except that we modify the respective data trees as follows: Each edge $\langle u, w\rangle$ is replaced by two edges $\langle u, v\rangle, \langle v, w\rangle$, where v is a new node. Now, to make a *Following*-step between two nodes corresponding to original tree nodes, we can use the relation

$$\begin{aligned}\mathit{Following}''(x,y) := \exists z_1 \exists z_2 \exists z_3\ \mathit{Child}^+(z_1,x) \wedge \\ \mathit{NextSibling}^\alpha(z_2,z_3) \wedge \mathit{Child}^+(z_3,y).\end{aligned}$$

where α is "1" for τ_{12}, "+" for τ_{13}, and "∗" for τ_{14}. □

THEOREM 5.8. *Conjunctive queries over the signatures*

$$\begin{aligned}\tau_{15} &:= \langle \mathit{Following}, \mathit{NextSibling}, (\mathit{Label}_a)_{a\in\Sigma}\rangle,\\ \tau_{16} &:= \langle \mathit{Following}, \mathit{NextSibling}^+, (\mathit{Label}_a)_{a\in\Sigma}\rangle,\\ \tau_{17} &:= \langle \mathit{Following}, \mathit{NextSibling}^*, (\mathit{Label}_a)_{a\in\Sigma}\rangle\end{aligned}$$

are NP-complete w.r.t. query complexity.

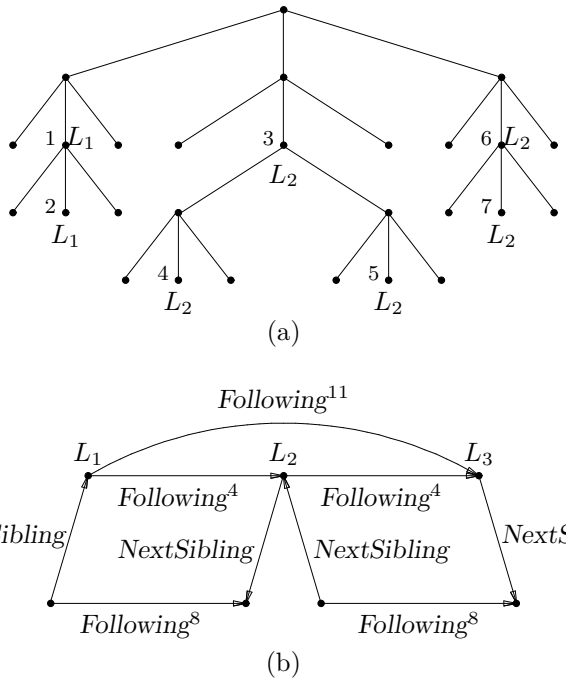


Figure 6: Encoding the selection of exactly one of the positive literals P, Q, R from clause $C = \{P, Q, R\}$ as a conjunctive query over signature τ_{15}.

Proof Sketch. We first look at signature τ_{15}. Consider the data tree shown in Figure 6 (a) and the query of Figure 6 (b).

As in the proof of Theorem 5.2, there is again one variable per label L_1 (L_2, L_3), which we call v_1 (v_2, v_3). Again, at most one variable v_1, v_2, and v_3 can be mapped to its topmost position. The query shown in Figure 6 (a) requires that precisely the partial assignments

$$\begin{aligned}&\theta(v_1) := 1,\ \theta(v_2) := 4,\ \theta(v_3) := 7\\ &\theta(v_1) := 2,\ \theta(v_2) := 3,\ \theta(v_3) := 7\\ &\theta(v_1) := 2,\ \theta(v_2) := 5,\ \theta(v_3) := 6\end{aligned}$$

can be extended to solutions of query.

This provides us with an encoding for the selection of exactly one literal from a given clause with three positive literals. The full reduction can be obtained analogously to the previous proofs to yield an encoding for 1-in-3 3SAT over positive literals.

The same reduction can be used to prove the corresponding result for the signatures τ_{16} and τ_{17}. □

6. EXPRESSIVENESS

Let F be a set of XPath axes and let F^{-1} denote their inverses (e.g., *Parent* for *Child*, see [28]). We denote by CQ[F] the conjunctive queries over signatures $\langle (\mathit{Label}_a)_{a\in\Sigma}, F\rangle$. By PQ[$F$] we denote the positive (first-order) queries (written as finite unions of conjunctive queries) over F. We denote the acyclic positive queries over F by APQ[F]. [8]

[8]We lack the space to formally introduce XPath, but it is easy to show that for the positive, navigation-only fragment of XPath (called positive Core XPath in [15]), for some set F of axes, APQ[F] captures positive Core XPath[$F \cup F^{-1}$] on trees in which each node has (at most) one label.

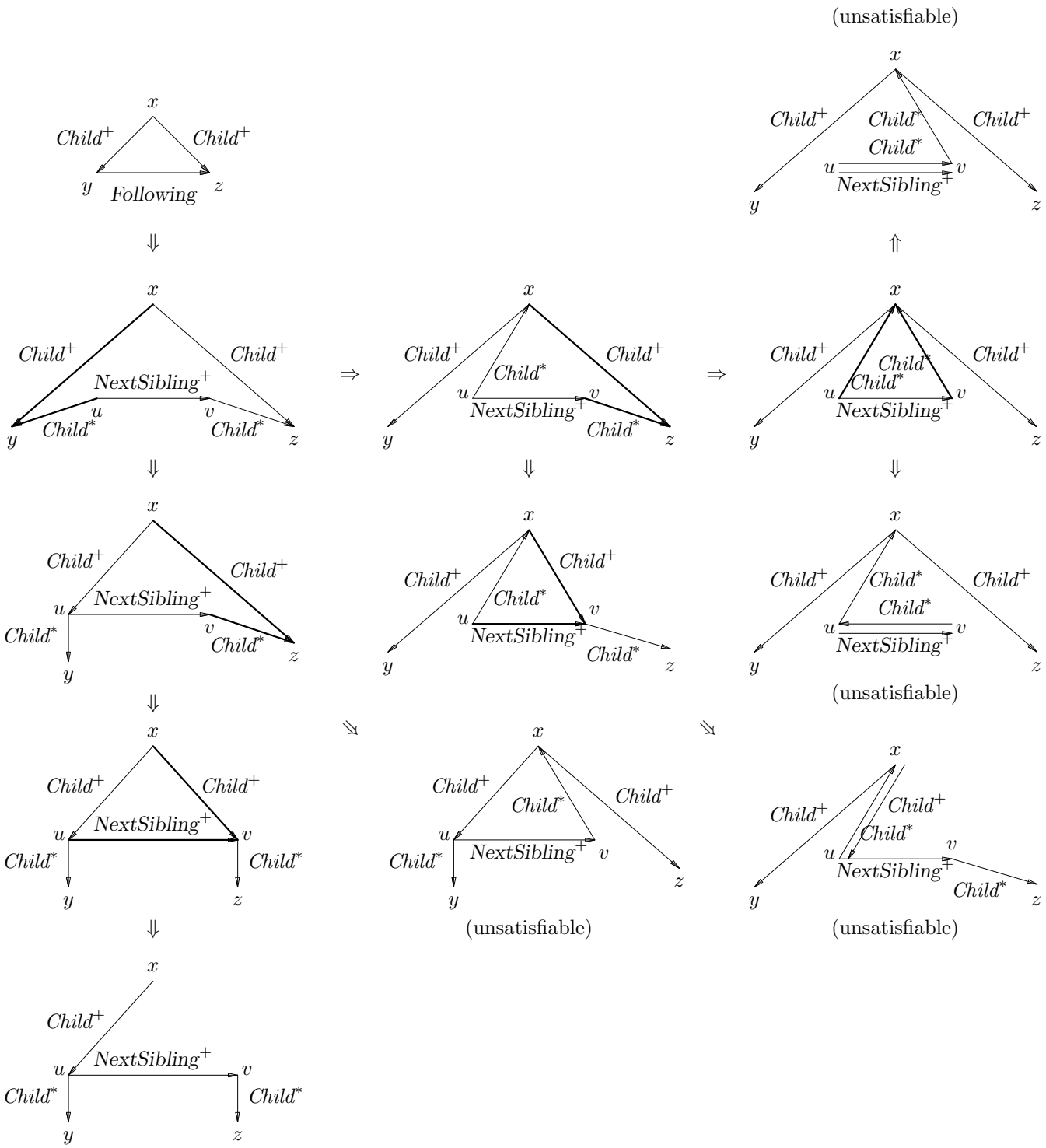


Figure 7: Translation of a conjunctive query into an APQ.

THEOREM 6.1. *Let F be a set of XPath axes. Then,*

$$CQ[F] \subseteq APQ[F].$$

Proof (Rough Sketch). We first rewrite all occurrences of *Following* using *Child** and *NextSibling*$^+$.

Then we proceed as follows. If there is a directed cycle of the query qraph that consists exclusively of *Child** and *NextSibling** edges, we eliminate it by unifying all variables in the cycle. We proceed analogously with undirected cycles which ensure that all variables in the cycle may be unified, such as *Child*$^*(x,y)$, *NextSibling*(x,y). If there is another kind of directed cycle, the query is unsatisfiable and we are done. There are a few further cases of undirected cycles that entail unsatisfiability and which are easy to detect, such as *Child*$^*(x,y), R(x,y)$, where R is either *NextSibling* or *NextSibling*$^+$. If there is no directed but an undirected cycle, the query contains two atoms $R_1(x,y)$, $R_2(x',y)$. Now, for any combination of

$$R_1, R_2 \in \{\textit{Child}, \textit{Child}^*, \textit{Child}^+, \textit{NextSibling}, \textit{NextSibling}^*, \textit{NextSibling}^+\},$$

we can rewrite these two atoms into at most two alternative atoms such that the cycle gets smaller and moves up or left in the tree. For example,

- if $R_1 = \textit{Child}$ and $R_2 = \textit{Child}$ or $R_1 = \textit{NextSibling}$ and $R_2 = \textit{NextSibling}$, we set $x = x'$.
- if R_1 is *Child* or *Child*$^+$ and R_2 is *NextSibling*, *NextSibling**, or *NextSibling*$^+$, we rewrite the two atoms into
$$R_1(x,x'), R_2(x',y).$$
- if $R_1 = \textit{Child}$ and $R_2 = \textit{Child}^+$, we rewrite the two atoms into *Child*(x,y), *Child*$^*(x',x)$
- if $R_1 = \textit{NextSibling}$ and $R_2 = \textit{NextSibling}^+$, we rewrite the two atoms into *NextSibling*(x,y), *NextSibling*$^*(x',x)$.
- if $R_1 = \textit{Child}$ and $R_2 = \textit{Child}^*$, we rewrite the two atoms either into *Child*(x,y), *Child*$^*(x',x)$ or set $x' = y$. (This means, in such a case we have to produce two alternative rewritings and continue rewriting both.)
- if $R_1 = \textit{Child}^*$ and $R_2 = \textit{Child}^*$, we rewrite the two atoms either into
$$\textit{Child}^*(x,y), \textit{Child}^*(x',x)$$
or into
$$\textit{Child}^*(x,x'), \textit{Child}^*(x',y).$$

(The remaining cases are obtained analogously.)

We repeat this (recursively for both alternative rewritings) until we either obtain an unsatisfiable cycle or a tree. In both cases we are done. The result is the union of the conjunctive queries that we compute and for which we do not infer unsatisfiability.

Note that the rewrite system is confluent and the ordering in which we choose pairs of binary atoms to rewrite does not matter. □

Therefore, each $CQ[F]$ query can also be formulated as an XPath query over axes $F \cup F^{-1}$. Using the rewriting technique sketched in the previous proof, each conjunctive query can be rewritten into a singly exponentially-sized APQ.

EXAMPLE 6.2. Figure 7 illustrates the translation of the previous proof by means of an example (the example query Q from the introduction). All conjunctive queries that we obtain are unsatisfiable, except for one. Thus, for Q there is an equivalent acyclic *conjunctive* query.

Note that in Figure 7 we make an exception by labeling the nodes of the query graph with the variable names in order to allow for the variables to be tracked through the rewrite steps more easily. □

Similar techniques to the one of the previous proof were used in [23] to eliminate backward axes from XPath expressions and in [27] to rewrite first-order queries over trees given by certain regular path relations.

COROLLARY 6.3. *PQ[F] = APQ[F].*

Obviously, the CQ[F] are not closed under union. On trees of one node only, conjunctive queries are equivalent to ones which do not use binary atoms. It is easy to see that the query $\{x \mid A(x) \vee B(x)\}$ has no conjunctive counterpart.

PROPOSITION 6.4. *CQ[F] ≠ APQ[F].*

Now, there are signatures for which all conjunctive queries can be rewritten into APQ's in polynomial time and there are signatures for which this is not possible.

PROPOSITION 6.5 ([13]). *Every CQ[Child, NextSibling] can be rewritten in linear time into an equivalent acyclic CQ[Child, NextSibling, NextSibling*].*

It is easy to verify by inspecting the proof in [13] that rewriting each CQ[*Child*, *NextSibling*] into an equivalent acyclic CQ[*Child*, *NextSibling*] in linear time is also possible. (The proof there also deals with relations such as *FirstChild*. If these are not present, *NextSibling** is not required.)

Finally, we show that there are conjunctive queries over trees that cannot be polynomially translated into equivalent APQs.

Let D_n denote the n-diamond query

$$D_n \leftarrow Y_1(y_1) \wedge \bigwedge_{i=1}^{n} \big(\textit{Child}^+(y_i,x_i) \wedge \textit{Child}^+(y_i,x_i') \wedge \textit{Child}^+(x_i',y_{i+1}) \wedge \textit{Child}^+(x_i',y_{i+1}) \wedge X_i(x_i) \wedge X_i'(x_i') \wedge Y_{i+1}(y_{i+1})\big).$$

A graphical representation of this Boolean $CQ[\{\textit{Child}^+\}]$ is given in Figure 8 (a).

THEOREM 6.6. *For D_n, no equivalent APQ over XPath axes exists that is of size polynomial in $|D_n|$.*

Proof. By contradiction.

Let A & B be a shortcut for the regular expression $(A.B \mid B.A)$ and let s be a shortcut for the path of 2^n unlabeled nodes. The regular expression

$$Y_1.s.(X_1.s \;\&\; X_1'.s).Y_2.s.(X_2.s \;\&\; X_2'.s).Y_3.s.\cdots \cdots .Y_n.s.(X_n.s \;\&\; X_n'.s).Y_{n+1}$$

defines 2^n path-structures (i.e., trees that are paths) over alphabet $\Sigma = \{X_1, \ldots, X_n, X_1', \ldots, X_n', Y_1, \ldots, Y_{n+1}\}$, as sketched in Figure 8 (b).

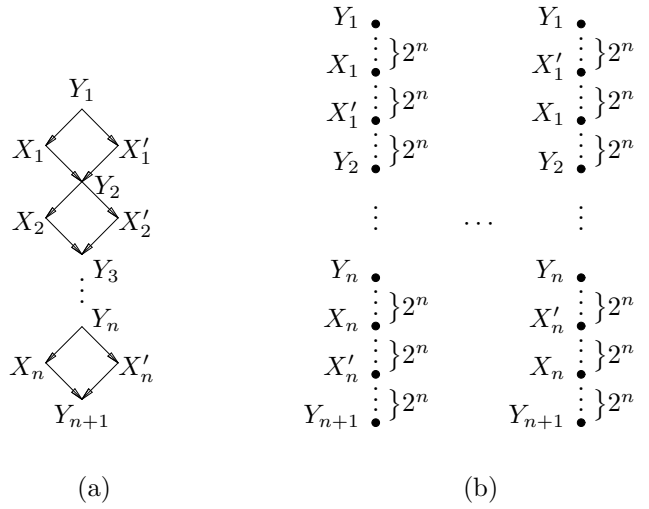


Figure 8: Query D_n (a) and path models (b) from proof of Theorem 6.6.

Assume there is a polynomially-sized (Boolean) APQ $Q = Q_1 \vee \cdots \vee Q_m$ which is equivalent to D_n. Then there is (at least) one subquery Q_i which returns true for exponentially many of our path-structures (let $\mathcal{P}$ denote this set of structures) and which does not return true for any structure for which D_n does not return true.

We only need to consider the *Child*$^+$ axis for binary relations in Q_i. Indeed, for each model $\mathcal{A}$ of D_n and each valuation θ, θ will map all variables of D_n into a single path in $\mathcal{A}$. Thus, if Q_i uses either the *NextSibling* or *NextSibling*$^+$ axis, Q_i can be dropped from Q without changing it (i.e., $Q \equiv Q - Q_i$). Similarly, each occurrence of an atom *NextSibling*$^*(x, y)$ entails that x and y must match the same node. Such an atom can be removed and all occurrences of y in Q_i be replaced by x. Since in our models, any two nodes that may match a variable from D_n (because of the unary predicates that can be matched at these nodes) are only reachable through a path of length at least 2^n, we cannot build a path between two variables from D_n using *Child* atoms, nor can we define a relationship between the two variables that we cannot define using just *Child*$^+$. (As required above, the nodes along the paths of length 2^n are unlabeled.) By similar considerations, it becomes clear that *Child** is not necessary either.

Consider the set of paths $\Pi \subseteq Var(Q_i)^*$ from nodes with in-degree zero to nodes with out-degree zero that occur in the query graph of Q_i. Since Q_i is acyclic, the number of such paths is bounded by the square of the number of its nodes. Since Q_i is moreover of polynomial size, there are only polynomially many paths.

Given a path $\pi \in \Pi$, let $l(\pi)$ denote the path of labels (unary atoms) associated to the path π of variables. (Since in our path-structures in $\mathcal{P}$, each node has at most one label, we may assume the same for π, otherwise Q_i is always false on the path models. We supplement "_" for variables which do not have a unary atom associated.) Let $l(\Pi) = \{l(\pi) \mid \pi \in \Pi\}$.

There are exponentially many distinct paths over

$$X_1, X'_1, \ldots, X_n, X'_n$$

to be matched in our set of path-structures $\mathcal{P}$. Whenever there is a path $u.X_j.v.X'_j.w$ or $u.X'_j.v.X_j.w$ in $l(\Pi)$ (where u, v, and w are words), an order between X_j and X'_j is fixed which must be reflected in all path-structures in $\mathcal{P}$, because in each path-structure precisely one node is labeled X_j and precisely one (different) node is labeled X'_j. To match exponentially many path-structures as we required for Q_i, there must be sufficiently – polynomially – many indexes j for which no such order between X_j and X'_j is determined.

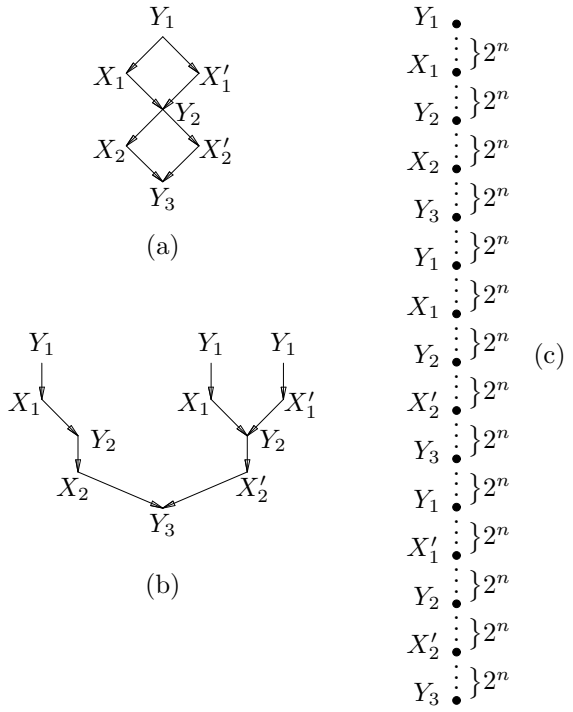


Figure 9: Example of the path structure construction of the proof of Theorem 6.6.

To simplify the presentation, we may now assume that there is no pair of variables for which this order is determined by Q_i. (Otherwise, there is a straightforward way to modify the queries Q and D_n and the path-structures of $\mathcal{P}$ to get to an equivalent problem where this is the case.) It follows that in no path in $l(\Pi)$, both a label X_j and the corresponding X'_j may occur.

Since there are only polynomially many paths in Π, there is a path in $(X_1 \mid X'_1).\cdots.(X_n \mid X'_n)$ which does not occur in $l(\Pi)$. W.l.o.g., let this path be $X'_1.X'_2.\cdots.X'_n$.

We now construct a path-model M of Q_i which is not a model of D_n. M is obtained by concatenating the paths in Π in a special order which is obtained as follows. Let $\Pi(X)$ (resp., $\Pi(\neg X)$) denote the set of paths in $l(\Pi)$ which contain label X (resp., do not contain label X). Let $\Pi(\phi \wedge \psi) = \Pi(\phi) \cap \Pi(\psi)$. As a shortcut, let $\phi_k = \bigwedge_{i=1}^{k}(\neg X_i \wedge X'_i)$. Now, we define M as the path

$$\begin{aligned} M = \Pi(X_1).\Pi(\neg X_1 \wedge \neg X'_1).& \\ \Pi(\phi_1 \wedge X_2).\Pi(\phi_1 \wedge \neg X_2 \wedge \neg X'_2).&\cdots \\ \Pi(\phi_{n-1} \wedge X_n).\Pi(\phi_{n-1} \wedge \neg X_n \wedge \neg X'_n).& \\ \Pi(\phi_{n-1} \wedge \neg X_n \wedge X'_n)& \end{aligned}$$

The paths in each $\Pi(\psi)$ may be concatenated in any order.

Consider for example the acyclic conjunctive query Q' of Figure 9 (b). Here, the only path of the 2-diamond query D_2 shown in Figure 9 (a) which does not occur in Q' is $Y_1.X_1'.Y_2.X_2.Y_3$. The path-structure M_5 constructed as described above is shown in Figure 9 (c). It consists of a concatenation of the two paths $Y_1.X_1.Y_2.X_2.Y_3$ and $Y_1.X_1.Y_2.X_2'.Y_3$ – which do contain X_1 (and which we can add to M_5 in any order) – and the path $Y_1.X_1'.Y_2.X_2'.Y_3$, which contains X_1' rather than X_1 and which is therefore added to M_5 after the other two paths. It is easy to see that indeed Q' is true on M_5, but that D_2 is not (the unique occurrence of X_1' in M_5 is below the unique occurrence of X_2).

M is indeed a model of Q_i. In fact, any concatenation of the paths in $l(\Pi)$ would satisfy Q_i. We define a valuation θ as follows. Let $\pi_1.x.\pi_1', \ldots, \pi_k.x.\pi_k'$ be the set of all paths in Π in which x occurs. Then,

$$\theta(x) = v \quad \Leftrightarrow \quad \begin{array}{l} v \text{ is the topmost node in } M \text{ s.t.} \\ \pi_1.x, \ldots, \pi_k.x \text{ can be matched in the} \\ \text{path from the root of } M \text{ to } v. \end{array}$$

Since each of the paths in $l(\Pi)$ occurs in M, θ is a complete function, i.e. defined for all variables in $Var(Q_i)$.

The valuation θ is also consistent. Assume that $\theta(x) = v$, y is a variable that occurs in a path of the form $\pi.y.\pi'.x.\pi'' \in \Pi$ (i.e., y occurs above x in the query graph), and $\theta(y) = w$. By definition, w is the topmost node for which all paths with a prefix $\pi.y$ can be matched in the subpath of M from the root to w. For each such $\pi.y$, $\pi.y.\pi'.x$ must match the path from the root of M to v. Thus, v must be below w in M.

On the other hand, M is not a model of D_n. The topmost node in M matching variable x_j' is never above the $\Pi(\phi_j)$-interval of the path. However, since no path with the ordering $X_1' \ldots X_2' \ldots X_3' \ldots \cdots \ldots X_{n-1}' \ldots X_n'$ exists in Q_i, the $\Pi(\phi_n)$-interval is empty.

This concludes our proof. □

7. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] R. Baumgartner, S. Flesca, and G. Gottlob. "Visual Web Information Extraction with Lixto". In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, 2001.

[3] M. Benedikt, W. Fan, and G. Kuper. "Structural Properties of XPath Fragments". In *Proc. of the 9th International Conference on Database Theory (ICDT'03)*, 2003.

[4] M. Bodirsky, D. Duchier, J. Niehren, and S. Miele. "A New Algorithm for Normal Dominance Constraints". In *Proc. SODA*, 2004.

[5] A. K. Chandra and P. M. Merlin. "Optimal Implementation of Conjunctive Queries in Relational Data Bases". In *Conference Record of the Ninth Annual ACM Symposium on Theory of Computing (STOC'77)*, pages 77–90, Boulder, CO USA, May 1977.

[6] C. Chekuri and A. Rajaraman. Conjunctive Query Containment Revisited". In *Proc. of the 6th International Conference on Database Theory (ICDT'97)*, pages 56–70, 1997.

[7] R. Dechter. *"Constraint Processing"*. Morgan Kaufmann, May 2003.

[8] A. Deutsch and V. Tannen. "MARS: A System for Publishing XML from Mixed and Redundant Storage". In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB'03)*, pages 201–212, 2003.

[9] A. Deutsch and V. Tannen. "Reformulation of XML Queries and Constraints". In *Proc. of the 9th International Conference on Database Theory (ICDT'03)*, pages 225–241, 2003.

[10] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1999. Second edition.

[11] J. Flum, M. Frick, and M. Grohe. "Query Evaluation via Tree-Decompositions". In J. Van den Bussche and V. Vianu, editors, *Proc. of the 8th International Conference on Database Theory (ICDT'01)*, volume 1973 of *Lecture Notes in Computer Science*, pages 22–38, London, UK, Jan. 2001. Springer.

[12] G. Gottlob and C. Koch. "Monadic Queries over Tree-Structured Data". In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 189–202, Copenhagen, Denmark, July 2002.

[13] G. Gottlob and C. Koch. "Monadic Datalog and the Expressive Power of Web Information Extraction Languages". *Journal of the ACM*, **51**(1):74–113, 2004.

[14] G. Gottlob, C. Koch, and R. Pichler. "Efficient Algorithms for Processing XPath Queries". In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*, Hong Kong, China, 2002.

[15] G. Gottlob, C. Koch, and R. Pichler. "The Complexity of XPath Query Processing". In *Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'03)*, 2003.

[16] J. Hidders. "Satisfiability of XPath Expressions". In *Proc. DBPL*, 2003.

[17] LDC. "The Penn Treebank Project", 1999. http://www.cis.upenn.edu/∼treebank/home.html.

[18] D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.

[19] M. P. Marcus, D. Hindle, and M. M. Fleck. "D-Theory: Talking about Talking about Trees". In *Proc. ACL*, pages 129–136, 1983.

[20] H. Meuss and K. U. Schulz. "Complete Answer Aggregates for Tree-like Databases: A Novel Approach to Combine Querying and Navigation". *ACM Transactions on Information Systems*, **19**(2):161–215, 2001.

[21] H. Meuss, K. U. Schulz, and F. Bry. "Towards Aggregated Answers for Semistructured Data". In *Proc. of the 8th International Conference on Database Theory (ICDT'01)*, pages 346–360, 2001.

[22] M. Minoux. "LTUR: A Simplified Linear-Time Unit Resolution Algorithm for Horn Formulae and Computer Implementation". *Information Processing Letters*, **29**(1):1–12, 1988.

[23] D. Olteanu, H. Meuss, T. Furche, and F. Bry. "Symmetry in XPath". In *Proc. EDBT Workshop on XML Data Management*, 2002.

[24] T. Schaefer. "The Complexity of Satisfiability Problems". In *Proc. 10th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 216–226, 1978.

[25] M. Schmidt-Schauss and K. U. Schulz. "On the Exponent of Periodicity of Minimal Solutions of Context Equations". In *Proc. 9th Int. Conf. on Rewriting Techniques and Applications*, pages 61–75, 1998.

[26] M. Schmidt-Schauß and J. Stuber. "On the Complexity of Linear and Stratified Context Matching Problems", 2001. Unpublished manuscript.

[27] T. Schwentick. "On Diving in Trees". In *Proc. MFCS*, pages 660–669, 2000.

[28] World Wide Web Consortium. XML Path Language (XPath) Recommendation. http://www.w3c.org/TR/xpath/, Nov. 1999.

[29] M. Yannakakis. "Algorithms for Acyclic Database Schemes". In *Proceedings of the 7th International Conference on Very Large Data Bases (VLDB'81)*, 1981.